Ep No: 1
22.02.2021

Simple Hill Climbing Problem

Srinidhi.V
18C102

Aim:
To perform simple hill climbing search algorithm

Algorithm:

Step 1: It will evaluate the initial state

Conditions:

a) If it is found to be final state, stop and return success

b) If it is not found to be the final state, make it a current state.

Step 2: If no state is found giving a solution, perform looping.

1. A state which is not applied should be selected as the current state and with the help of this state, produce a new state.

2. Evaluate the new state produced.

Conditions:

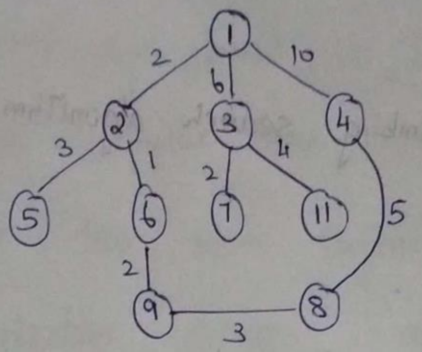
a) If it is found to be final state, stop and return success.

b) If it is found better compared to current state, then declare itself as a current state and proceed.

c) If it is not better, perform looping until it reaches a solution

Step 3: Exit the process.

Sample Input And Output:



Start Node → 1

Goal Node → 8

Current node → 1

2	3	4

cost = 2

Current node → 2

5	6

Cost = 3

Current node → 6

9

Cost = 5

Current node → 9

8

Cost = 8

Current node → 8

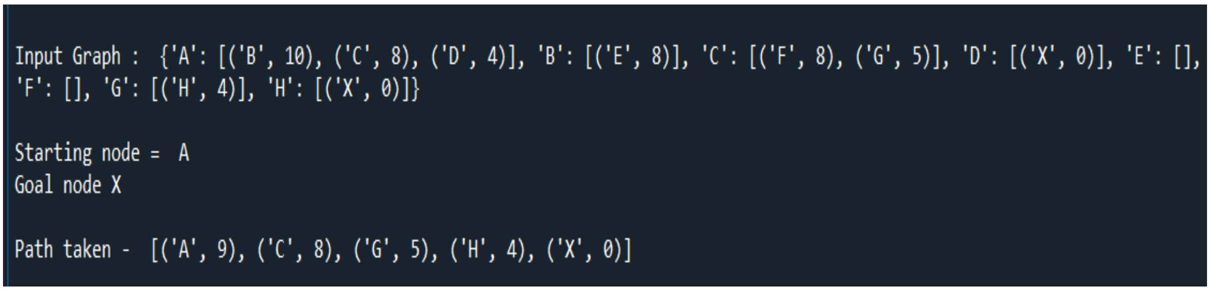
Goal reached

Cost of Path is 8

Program:

```
   def makeGraph():
    adj = dict()
    adj['A'] = [('B', 10), ('C', 8), ('D', 4)]
    adj['B'] = [('E', 8)]
    adj['C'] = [('F', 8), ('G', 5)]
    adj['D'] = [('X', 0)]
    adj['E'] = adj['F'] = []
    adj['G'] = [('H', 4)]
    adj['H'] = [('X', 0)]
    return adj


def hill_climbing(adj, curr_node, curr_val, goal):
    global path
    path.append((curr_node, curr_val))

    if(curr_node == goal):
        return

    target_node, target_val = '', 0
    for node, value in adj[curr_node]:
        if(value < curr_val):
            target_node = node
            target_val = value
            break

    hill_climbing(adj, target_node, target_val, goal)


if __name__ == "__main__":
    adj = makeGraph()
    print("\nInput Graph : ", adj)
    start, starting_val, goal = 'A', 9, 'X'
    print("\nStarting node = ", start)
    print("Goal node", goal)
    path = []
    hill_climbing(adj, start, starting_val,  goal)
    print("\nPath taken - ", path)
```

Output:



```
Input Graph :  {'A': [('B', 10), ('C', 8), ('D', 4)], 'B': [('E', 8)], 'C': [('F', 8), ('G', 5)], 'D': [('X', 0)], 'E': [],
'F': [], 'G': [('H', 4)], 'H': [('X', 0)]}

Starting node =  A
Goal node X

Path taken -  [('A', 9), ('C', 8), ('G', 5), ('H', 4), ('X', 0)]
```

Result:

Thus, the hill climbing problem is implemented successfully.